**Predictive Maintenance System Using**

**Machine Learning**

**Testing Plan and Test Scripts**

**Prepared By:** Tapan Soni

**Date:** April 10th, 2018

**Team Ostriches – Rowan University**

**Product Owner:** Craig Wert

**Scrum Master:** John Stranahan

**Development Team:** Tapan Soni, Joshua Jackson, Nicholas La Sala, Michael Matthews

**Table of Contents**

1. Project Overview……………………………………………………………………………………… 3

2. Validation Plan Overview…………………………………………………………………………. 3

3. Related Documents………………………………………………………………………………….. 4

4. Test Environment……………………………………………………………………………………. 4

4.a Requirements………………………………………………………………………………. 4

4.a.i Software………………………………………………………………………….. 4

4.a.i.1 Recommended……………………………………………………… 4

4.a.i.2 Minimum Requirements………………………………………… 4

4.a.ii Hardware……………………………………………………………………….. 5

4.a.i.1 Recommended……………………………………………………… 5

4.a.i.2 Minimum Requirements………………………………………… 5

4.a.iii Personnel………………………………………………………………………. 5

4.a.iii.1 Recommended……………………………………………………. 5

4.a.iii.2 Minimum Requirements………………………………………. 5

5. Test Scripts……………………………………………………………………………………………. 5 - 14

1. **Project Overview**

The Predictive Maintenance system uses machine learning to predict whether an aircraft engine needs to be repaired. The system uses the scikit-learn library in the python programming language to create a classifier. The classifier is trained on a provided dataset. To predict whether or not an engine needs to be repaired is done by using the K-Neighbor classifying algorithm which bases its decision on the nearest known data points. After normalizing the dataset and training the classifier on the training set and validating it on the validation data set, the system can make accurate predictions over all the rows of the data. After the prediction is made, the system returns either a ‘Yes, the engine needs to be repaired’, or ‘No, everything is OK’. This testing script is designed to test the functional requirements finished up till Sprint 4 of the project.

**2. Validation Plan Overview**

The GUI was created through the TKinter module because it was easy to program with and because TKinter can easily be integrated into Python 3.X versions. The GUI had to meet the requirements of showing the output after the prediction had been made, it had to show the input file which could only be a CSV, and it had to show the time of the entire run of the program and that of the classifier itself. These requirements were done with the Definition of Done in mind that both the Ostriches and ASRC had agreed to. The classifier had to be chosen from a list of supervised classifying algorithms such as: K-Nearest Neighbor, Decision Trees, Neural Nets, Gaussian Naïve Bayes, and Gradient Descent. The team conducted different experiments with each classifier, determining which classifier was the best in terms of accuracy, and simplicity in implementation. The team experimented with different tagging and data splitting ratios and finally settled on K-Nearest Neighbor as the best classifier. To increase the speed of the program, the classifier and the data set were serialized, so they could be loaded into the program very quickly. That reduced the runtime from about +1 minute to less than 23 seconds. All of these requirements were done adhering to the Definition of Done. As per the approval from one of the sponsors, Rukan Shao, the team is working on implementing the K-Cluster Unsupervised machine learning algorithm to test its capabilities and accuracy on the data to see if unsupervised machine learning would be a better match for the project, and give a better predictive algorithm than the supervised machine learning techniques.

**3. Related Documents**

Documents relating to the testing script are the Requirements Document, the Experiment Log, and the Validation and Traceability Matrix.

**4. Test Environment**

The testing environment will be in an Integrated Development Environment (IDE), and will consist of the following requirements:

**a. Requirements**

**I. Software**

The tester must have Python 3.6.X or higher installed on their system for this system to function correctly. The tester may have either a Python IDE (PyCharm Community 2017 edition or IDLE are the recommended IDEs), or have a method to run this program from the terminal and to direct test files into the program.

**1. Recommended**

Operating System: Windows 10, macOS, Linux

IDE: PyCharm Community 2017.3.3 or higher, IDLE, Anaconda 1.8.2 , or Terminal

Python Version: 3.6.5 or higher

Python Modules: scikit-learn v0.19.1, numpy v1.14.2, tkinter v8.6, scipy v1.0.1, math (built in library), random (built in library)

**2. Minimum Requirements**

Operating System: Windows 7 or later, macOS, or Linux

IDE: PyCharm Community 2017.3.1 or higher

Python Version: 3.6.1 or higher

Python Modules: scikit-learn v0.19, numpy v1.13.9, tkinter v8.6, scipy v1.0, math (built in library), random (built in library), pickle (built in library)

**II. Hardware**

**1. Recommended**

Processor: Intel Core i5 4800M or faster processor.

Disk Space: 20GB

RAM: 8GB

**2. Minimum Requirements**

Processor: Intel Core i3 4130 or faster

Disk Space: 10GB

RAM: 4GB

**III. Personnel**

**1. Recommended**

The tester should have experience in Big Data Analytics, Machine Learning algorithms, and Machine Learning applications. The tester will also need to be able to clearly record and communicate the results of the tests to their teams. The tester should also be proficient in reading and analyzing source code in the IDE to verify the requirements.

**2. Required**

The tester must have experience in machine learning algorithms and their applications. The tester must have experience looking at source code and verifying procedures. The tester must be able to follow instructions to initiate and complete the tests, then record and communicate the results of the tests to their team members.

1. **Test Scripts**

The test scripts start on the next page – page 6.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 0 |  | 1 | Download PyCharm and load the project |  |  |  |  |  |
| 0 |  | 2 |  | Go to the IntelliJ website and download the PyCharm IDE Community Edition | <https://www.jetbrains.com/pycharm/download/#section=windows> | The IntelliJ PyCharm download website will load |  |  |
| 0 |  | 3 |  | Click the Download button under the Community version | Button Click | IntelliJ will start the download |  |  |
| 0 |  | 4 |  | Install and configure the IDE | Click Agree on the license acceptance and next to install the IDE | PyCharm will be installed |  |  |
| 0 |  | 5 |  | Open PyCharm | Double click the PyCharm Icon | PyCharm will open |  |  |
| 0 |  | 6 |  | Install the required packages | Go to the File > Settings > Project Interpreter. Click the + arrow on th right, and download Scikit-Learn, numpy, tkinter, random, scipy | PyCharm will install the selected libraries and the project is able to run |  |  |
| 1 | 3.1, 3.1.1, 3.1.4, 3.1.5, 3.1.2 |  | Test the GUI's aesthetics |  |  |  |  |  |
| 1 |  | 1 |  | Load the GUI file | Double click on the NewGUI.py file in the project file explorer | The NewGUI.py file will load in the editor |  |  |
| 1 |  | 2 |  | Run the file | Right click in the editor | The right click menu will appear |  |  |
| 1 |  | 3 |  | Run the file | Click on the "Run NewGui.py" button | The NewGUI.py file will run and the GUI will appear |  |  |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 1 |  | 4 |  | Confirm that the product has a GUI (REQ - 3.1) | N/A | The product has a GUI |  |  |
| 1 |  | 5 |  | Confirm that the GUI has a title bar (REQ - 3.1.1) | N/A | The GUI has a title bar |  |  |
| 1 |  | 2 |  | Run the file | Right click in the editor | The right click menu will appear |  |  |
| 1 |  | 6 |  | Confirm that the GUI has a run button and browse button - they are combined into one button (REQ - 3.1.4, 3.1.2) | N/A | The GUI has a run button |  |  |
| 1 |  | 7 |  | Confirm that the GUI has an output area for the output - the white label at the bottom (REQ - 3.1.5) | N/A | The GUI has an output section for the output |  |  |
| 2 | 3.1.3, 3.2.1, 3.2.3, 3.2.4, 3.3 |  |  | Test the GUI functionality and output |  |  |  |  |
| 2 |  | 1 |  | Load the GUI File | Double click on the NewGUI.py file in the project file explorer | The NewGUI.py file will load in the editor |  |  |
| 2 |  | 2 |  | Run the file | Right click in the editor | The right click menu will appear |  |  |
| 2 |  | 3 |  | Run the file | Click on the "Run NewGui.py" button | The NewGUI.py file will run and the GUI will appear |  |  |
| 2 |  | 4 |  | Load the test data file | Click on the "Browse and Run" Button | The file browser will pop up |  |  |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 2 |  | 5 |  | Select the file and run the program | Once the browse window pops up, select the TestData.csv file from where you saved it and click open | The program will accept the CSV test data and the file browser window will close and the program will start to run |  |  |
| 2 |  | 6 |  | Confirm that the only available option for files is a .CSV file (REQ - 3.2.1, 3.3) | N/A | The only type of data that can be input is of .CSV extension |  |  |
| 2 |  | 7 |  | Wait for the program to finish | Wait | The program will finish running and print its output in the output window. The timestamps will also be printed on the GUI. The file chosen will also be displayed in the file name display |  |  |
| 2 |  | 8 |  | Confirm that the program has output a "No - Everything is OK" (REQ - 3.2.4) | N/A | The program has output "No - Everything is OK" in the output area under the "Browse & Run" button |  |  |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 2 |  | 9 |  | Confirm that the program has at least a 50% accuracy (REQ - 3.2.3) | Look at the number after K-Neighbor in the console. | It should be grater than 0.50 |  |  |
| 3 |  | 1 |  | Look at the Kneighbor.py File | Double click on the Kneighbor.py file from the project explorer | The Kneighbor.py file should come into the editor |  |  |
| 3 | 3.2, 3.2.5 |  | Predictive Algorithm |  |  |  |  |  |
| 3 |  | 2 |  | Look at the Kneighbor.py File | Look at line # 31 (± 2 lines) | You should see the import of the K Neighbor classifier from the sklearn package |  |  |
| 3 |  | 3 |  | Confirm that the program uses the predictive algorithm (REQ - 3.2, 3.2.5) | N/A | The program uses the K Neighbor classifier to predict over a certain number of rows (testdata.CSV) and from an average of the output (50% or higher of the predictions are 0 - Good, Less than 50 - Maintenance needed) |  |  |
| 4 | 3.2.6 |  | Serialized Classifier |  |  |  |  |  |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 4 |  | 1 |  | Look at the NewGUI.py file | Double click on the NewGUI.py file | The NewGUI.py file loads into the editor |  |  |
| 4 |  | 2 |  | Look at the NewGUI.py file | Look at line 240 (± 5 lines) | The pickled file (the serialized file) will be opened from its serialized format (classy.pkl) into the variable pkl\_file |  |  |
| 4 |  | 3 |  | Look at the NewGUI.py file | Look at line 243 (± 5 lines) | The pkl\_file will be unpickled (unserialized) and assigned to the variable classy which is what is going to be used in predicting over the test data |  |  |
| 4 |  | 4 |  | Look at the NewGUI.py file | Look at line 256 (± 5 lines) | The classy variable (the unserialized classifier) will be used to predict over each line in the testData.csv file. |  |  |
| 4 |  | 5 |  | Run Test # 2 | Follow the directions for Test # 2 to verify that the product works with the testData.csv input | The program will output a "No - Everything is OK" and end |  |  |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 4 |  | 6 |  | Confirm that the program uses the serialized classifier (REQ - 3.2.6) | N/A | The program uses the serialized classifier |  |  |
| 5 | 3.2.2 |  | The application needs to be more accurate |  |  |  |  |  |
| 5 |  | 1 |  | Look at the experimental log | Go to https://docs.google.com/spreadsheets/d/1N\_rmUx13Gc9WhWzIMXTTJss4tS2L5ILz2r0u0C5BHbI/edit?usp=sharing | You should be directed to the SWENG experimental log |  |  |
| 5 |  | 2 |  | Look at the experimental log | Look at the experimental log | The experimental log is sorted from the highest accuracy to the lowest accuracy and each attribute of the particular test that resulted in that accuracy is also listed |  |  |
| 5 |  | 3 |  | Confirm that the application has become more accurate (REQ - 3.2.2) | N/A | The classifier has become more accurate over the numerous experimental runs. The tagging system of70 – 30 yields an accuracy of above 90% |  |  |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
| 6 | 3.1.6 |  | Verify that the About Button works |  |  |  |  |  |
| 6 |  | 1 |  | Load the NewGUI.py file | Double click on the NewGUI.py file | The NewGUI.py file should populate in the editor |  |  |
| 6 |  | 2 |  | Run the file | Right click in the editor | The right click menu will appear |  |  |
| 6 |  | 3 |  | Run the file | Click on the "Run NewGui.py" button | The NewGUI.py file will run and the GUI will appear |  |  |
| 6 |  | 4 |  | Click the about button | Click on the "About" button | A new window will pop up with the title "About" |  |  |
| 6 |  | 5 |  | Check the contents | N/A | The about window will have the basic instructions to use the program and other important information pertaining to the program |  |  |
| 6 |  | 6 |  | Confirm that the "About" Button works (REQ - 3.1.6) | N/A | The about window will pop up with the basic instructions and other information |  |  |
| **COULD NOT TEST** | | | | | | | | |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
|  | 3.4 |  | All analyses run on data should be logged for later review in case it is deemed necessary |  |  |  |  |  |
|  | 4.1.1 |  | The application must be secure, as it is for military use |  |  |  |  |  |
|  | 4.1.2 |  | The application must be username and password protected |  |  |  |  |  |
|  | 4.1.3 |  | All login attempts must be recorded |  |  |  |  |  |
|  | 4.1.4 |  | Only authorized personnel should have access to the data used with this application |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COULD NOT TEST** | | | | | | | | |
| Test ID | Req Doc ID | Step | Test Description | Procedure | Input | Expected Result | Actual Output | Pass or Fail (P/F) |
|  | 4.1.5 |  | Only personnel with high clearance and the maintenance crew should be allowed a login to this application |  |  |  |  |  |
|  | 4.1.6 |  | If the application communicates with a network, the network should be protected and/or local to ensure retention of vital data |  |  |  |  |  |